Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Angelina Temelkovska

Primerjava atributov za napovedovanje ocen v spletni trgovini Amazon

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Lovro Šubelj

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu primerjajte napovedno moč različnih atributov za napovedovanje številčnih ocen produktov v spletni trgovini Amazon. V analizo vključite atribute kot so časovne značke ocen, koristnost, besedilne ocene, interakcijo med uporabniki in drugo. Za napovedovanje uporabite izbrane pristope odkrivanja znanj iz podatkov in besedil ter pa metode analize kompleksnih omrežij. Primerjajte uspešnost izbranih pristopov, kritično ovrednotite rezultate ter podajte predloge za nadaljnje delo.

Zahvaljujem se svojemu mentorju, doc. dr. Lovru Šublju, za vso pomoč pri izbiri teme diplomske naloge, za dostopnost, potrpežljivost in podane smernice za pisanje ter opogumljanje v trenutkih dvoma.

Svojim staršem se zahvaljujem za brezpogojno ljubezen, podporo, zaupanje in razvajanje. Hvala, ker me nista nikoli omejevala pri uresničevanju mojih želja in načrtov.

Najlepša hvala tudi Ireni in Bojanu, ki sta me iskreno toplo sprejela v svoj dom, tako sem svojega manj pogrešala.

Hvala tudi vsem kolegicam in kolegom, prijateljicam in prijateljem, ki delijo z menoj trenutke svojega življenja in s tem polepšajo ter obogatijo tudi mojega.

На Лилјана, Митко и Филип,
таму каде што вечно сонцето сјае.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
LR	linear regression	linearna regresija
SVM	support vector machine	metoda podpornih vektorjev
RF	random forest	naključni gozdovi
MAE	mean absolute error	povprečna absolutna napaka
RMSE	root mean square error	koren povprečne kvadratne napake
R^2	coefficient of determination	koeficient determinacije
tf–idf	term frequency – inverse document frequency	transformacija uteži inverzne frekvence besede
BC	betweenness centrality	središčnost vmesnosti
CC	closeness centrality	bližinska središčnost
DC	degree centrality	središčnost stopnje vozlišča
EVC	eigenvector centrality	središčnost lastnega vektorja
PR	PageRank	uvrstitev strani
HITS	hubs and authorities	kazala in viri
AUTH	authorities	viri
HUBS	hubs	kazala

Povzetek

Naslov: Primerjava atributov za napovedovanje ocen v spletni trgovini Amazon

Vsak dan komentiramo ali ocenjujemo stvari, ki nas obkrožajo in s katerimi se ukvarjamo, pri tem pa za seboj puščamo sledi. Cilj diplomske naloge je prikazati, kako napovedovati ocene v dandanes največji spletni trgovini Amazon. V ta namen smo zgradili modele v treh različnih fazah, in sicer iz treh različnih vendar povezanih področij analize podatkov.
Na začetku so prikazani matematični in statistični modeli, ki jih bomo uporabili za dosego cilja, in kratek pregled vsakega področja. Preko modelov bomo pokazali, kateri od atributov spletnih ocen in mnenj nam ponujajo največ informacij za številčno oceno.
Posamezni pristopi omogočajo pridobivanje različnih lastnosti, kot so časovni atribut, koristnost, besede v komentarjih in interakcija med uporabniki. V vsaki fazi nato preiskujemo dobljene lastnosti preko primerjanj različnih kombinacij morebitnih lastnosti. V nadaljevanju njihovo uspešnost ocenjujemo preko napovedovanja ocen.
V zaključku so predstavljene prednosti in slabosti zgrajenih modelov ter navedene možnosti za izboljšave in nadaljno delo v prihodnosti.

Ključne besede: strojno učenje, podatkovno rudarjenje, rudarjenje besedil, analiza omrežij, filmi, ocene, komentarji, uporabniki.

Abstract

Title: Comparison of attributes for predicting online reviews on Amazon

On every day basis, we grade and make comments about many subjects around us. This thesis is aiming to show how can we predict the grades on the largest online retailer nowadays Amazon. For that purpose, we built models in three different phases, by three different but also closely connected fields in the data analysis branch.
At the beginning, we give a short overview of each field and basic mathematical description of the models and estimators we use. Via those models, we show the big picture of which review's attributes give us the most information about user's numerical score.
Each of the three approaches extracts various attributes such as the time stamp, the helpfulness, the words in the comments or the interaction between the users, to name a few. Furthermore, we explore those features by comparing different combinations of them at each of the three steps. Then, we evaluate their success in making a prediction of the numerical score in each review.
At the end, we conclude with some of the advantages and disadvantages of the built models and possibilities for future improvements and further work.

Keywords: machine learning, data mining, text mining, network analysis, movies, scores, comments, users.

Poglavje 1

Uvod

Delovanje današnjega spleta, kot ga poznamo, ne bi bilo smiselno brez medsebojne interakcije uporabnikov. V obdobju, ko lahko podatki in informacije precej prispevajo k razumevanju te interakcije, se pokaže pomen strojnega učenja, saj je to področje, ki nam lahko pomaga izkoristiti te podatke in s tem pridobiti določena znanja. Najprej se ponudi možnost za interpretacijo tega znanja z različnih vidikov, ki se trenutno zdijo smiselni ali uporabni. Bistvena naloga pa je določiti, kateri del pridobljene informacije nam prinaša največjo korist.

V duhu potrošniške družbe je elektronsko poslovanje eno izmed bolj priljubljenih področij, kjer lahko znanje o nakupih prinese tudi dodatno dobičkonosno vrednost. Da bi to znanje uspešno pridobili, je potrebno pregledati in preučiti vedenje preko sledi, ki jih uporabniki največkrat pustijo za seboj v obliki ocen in komentarjev.

Sodobne spletne trgovine kot so *eBay* in *Amazon* ter spletne strani s podobnim značajem uporabnikom omogočajo svobodo v izražanju svojih lastnih mnenj in izkušenj, ki so vezani na izdelke. Tovrstno ocenjevanje nato pomaga ostalim uporabnikom pri odločanju o nakupu določenega izdelka. Poleg tega je uporabnikom omogočeno tudi ocenjevanje ocen oziroma izražanje mnenja o tem, koliko jim je določena ocena pomagala.

Cilj diplomske naloge je prikazati en del takšne interakcije med uporab-

niki in izdelki: z enostavno analizo o vsaki oceni izluščiti najbolj uporabne informacije, preko katerih bomo poskusili pridobiti najbolj natančno napoved ocen. Da bi opravili analizo, bomo izkoristili osnovne tehnike in metode strojnega učenja ter sorodnih področij. V prvih nekaj poglavjih je podan kratek povzetek teh področij, vključujoč matematične modele in metode, ki jih uporabljamo pri primerjavi. Temu sledi opis uporabljenih programskih orodij, in sicer najpomembnejših knjižnic in metod znotraj posamezne knjižnice.

Praktični del naloge je zasnovan v treh fazah, od katerih je vsaka naslednja kompleksnejša od prejšnje. Na ta način bo možno rezultate vseh treh faz medsebojno primerjati.

Prva faza zahteva, da izluščimo samo osnovne, primitivne atribute v takšni obliki kot so. Naslednji korak je enostavna frekvenčna obdelava besede vsakega mnenja, ki vključuje številčno oceno. Zadnja faza sestoji iz prepoznave atributov na podlagi grafov, ki so dobljeni preko števila uporabnikovih ocen. Preko izvedenih primerjav bomo ugotovili, kako uporabni so zgoraj omenjeni atributi. To lahko ocenimo preko napovedovalnih modelov in mer, ki nam sporočajo o napakah pri napovedovanju številčnih ocen.

V zadnjem poglavju združimo sklepe vsake faze in podano je mnenje glede izpolnitev pričakovanj. Predlagane so tudi možnosti za izboljšave in napotki za prihodnje delo, ki naj bi izboljšali zgoraj opisane faze.

Poglavje 2

Osnovni pojmi in metode

V tem poglavju je predstavljen pregled osnovnih treh področij, iz katerih izhaja ideja naloge. Sledi pregled metod oziroma regresijskih algoritmov, s pomočjo katerih je bil izveden praktični del diplomskega dela. Na koncu poglavja so podane meritve, preko katerih smo ocenjevali in primerjali rezultate posameznih algoritmov in področij.

2.1 Definicije in področja

Osnovna organizacijska enota, na kateri smo izvedli preiskovanje na podatkih, je primer, ki ga lahko zapišemo kot:

$$x_i = (x_1^i, x_2^i, x_3^i, \ldots, x_m^i), \tag{2.1}$$

kjer so posamezni elementi x_j^i vrednost j-tega atributa i-tega primera. Primer nam predstavlja najmanjši zapis v podatkovni množici, na kateri izvajamo operacije strojnega učenja.

Atribut je določena lastnost posameznega primera in ga najpogosteje predstavimo z diskretno ali zvezno vrednostjo. Razred je atribut, katerega vrednosti želimo napovedovati. Skupina primerov sestavlja množico primerov. To zapišemo kot matriko, ki kot vrstice vsebuje posamezne primere, stolpci pa predstavljajo atribute:

$$X_{m-1,n} = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_{m-1}^1 \\ x_1^2 & x_2^2 & \cdots & x_{m-1}^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_{m-1}^n \end{pmatrix}.$$

Atribut, ki mu je dodeljena vloga razreda, zapišemo kot transponiran vektor y_m:

$$y_m = (x_m^1, x_m^2, \ldots, x_m^n)^T, \tag{2.2}$$

in običajno se zapiše kot zadnji, m-ti stolpec, dodan matriki X.

2.1.1 Podatkovno rudarjenje

Potreba po sistematičnem izkoriščanju podatkov, ki so na razpolago, je svojo obliko počasi pridobivala preko par stoletij, vzporedno z razvojem matematičnih in statističnih modelov.

Področje podatkovnega rudarjenja (angl. *data mining)* nam omogoča prvotno obdelavo surovih podatkov [9, 16, 20, 29]. Termin obstaja približno od začetka 90-ih let 20. stoletja in se še vedno razvija. Vse bolj priljubljena postajata pojma *predictive analytics* in *data science*, vendar splošni pomen ostaja enak.

Z uporabo statističnih metod in metod strojnega učenja rudarjenje pripravi podatke v obliko, primerno za nadaljnjo analizo. Primer takšne oblike je zapis podatkov v obliki množice primerov iz uvodnega dela tega poglavja. Iz dane datoteke, kjer so zapisi v obliki slovarja, in sicer kot: {"kljuc": vrednost}, za vsak posamezen primer s pomočjo ključev pridobimo vrednosti atributov in to shranimo v nam ustrezni obliki, omenjeni v naslednjem poglavju.

Poleg tega podatkovno rudarjenje vključuje tudi že zgodnejšo ekstrakcijo podatkov, še preden so ti v obliki slovarja, najpogosteje neposredno iz baze podatkov. Vključuje pa tudi vizualizacijo podatkov in rezultatov ter posodabljanje sistemov, ki temeljijo na obdelavi podatkov v realnem času.

Podatkovno rudarjenje je tesno povezano s področjem strojnega učenja, saj meja med njima ni strogo ločena.

Rudarjenje besedil

Kot poseben aspekt podatkovnega rudarjenja omenjamo analizo besedil (angl. *text mining)*. Ta analiza je namreč ena izmed bolj kompleksnih modernih področij, saj je naravni jezik še vedno preveč kompleksen za računalnik. Dinamičnost naravnega jezika računalniku za zdaj preprečuje, da bi lahko dosegel zaželeno raven intuitivnosti in samostojnosti pri generiranju besedil ali prepoznavanju govora [22, 28].

Svoje mesto na tem področju pa najdejo analize z osnovnimi ter bolj naprednimi statističnimi metodami na podlagi frekvenc besed. Kot primer lahko omenimo priporočilni sistem, zasnovan za kronološko analizo uporabniških komentarjev [23].

V pričujoči nalogi se bomo dotaknili enostavne frekvenčne analize, in sicer preko frekvenc *tfidf*. Sestavljena je iz dveh delov, kot vidimo iz enačbe (2.3):

$$tfidf_{i,j} = tf_{i,j} \times \log \frac{N}{df_i}, \tag{2.3}$$

kjer je $tf_{i,j}$ je relativna frekvenca besede i znotraj posameznega dokumenta j, ki se potem uteži s frekvenco besede znotraj celega korpusa dokumentov, ki jih analiziramo. N predstavlja skupno število dokumentov, df_i pa število dokumentov, ki vsebujejo besedo i. Sledi kratek primer izračuna.

Beseda "dež" se v besedilu X, ki vsebuje 100 besed, pojavi trikrat. Pri tem pa celotni korpus Y šteje 10 000 besedil. 1000 tisoč od le-teh vsebuje besedo "dež" vsaj enkrat. Njena frekvenca znotraj besedila torej znaša:

$$tf = \frac{3}{100} = 0{,}3 \tag{2.4}$$

Inverzna frekvenca znotraj korpusa ustrezno dobi vrednost:

$$idf = \log \frac{10000}{1000} = log10 = 1 \tag{2.5}$$

Z množenjem obeh zgornjih rezultatov dobimo skupno uteženo frekvenco tfidf, ki znaša 0,3 in označuje vrednost tfidf za besedo "dež" znotraj besedila

X in korpusa Y.

Za lažjo manipulacijo podatkov v diplomski nalogi vsak posamezni primer predstavlja en dokument. Prednost izračuna teh frekvenc je v tem, da redke besede dobijo večjo inverzno težo idf in na ta način niso zanemarjene ob nadaljni analizi. Te besede so bolj pomembne, saj so bolj specifične.

2.1.2 Strojno učenje

Strojno učenje (angl. *machine learning)* [20] kot računalniško področje temelji na umetni inteligenci in predstavlja iskanje vzorcev v podatkih, z namenom, da na podlagi le-teh nekaj sklepali ali nekaj odkrili. Osnovna ideja je, da se modeli med izvajanjem sproti sami učijo. S tem dosežemo avtonomni in zanesljiv pristop analize, kjer ni potrebno ročno določati ter se prilagajati podatkom. Bistvena težava je, kako zastaviti problem, kako sestaviti pot do rešitve problema in pri tem izmed vseh možnosti izbrati čim bolj natančne metode.

Najprej je potrebno vedeti, ali so vrednosti razredov znane. V primeru, da to drži, govorimo o nadzorovanem učenju, saj vemo, da napovedujemo vrednost razreda. Če pa nam nabor razreda ni znan, govorimo o nenadzorovanem učenju, kjer je glavni cilj iskanje odvisnosti znotraj podatkov. Primer takšne metode je hierarhično razvrščanje, ki primere razdeli ali združi v skupine na podlagi določenega kriterija, kot je na primer najmanjša Evklidska razdalja [1] med primeri.

Nato je potrebno ugotoviti, ali je problem regresijski ali klasifikacijski. Pri prvem napovedujemo razred, ki ima zvezno zalogo vrednosti. Pri drugem pa napovedujemo diskretne vrednosti razreda oziroma ali primer pripada v razredu ali ne. Ta korak nam naprej določa, katere metode in katere meritvene napake bomo uporabljali, saj niso vsi enako primerni bodisi za regresijo bodisi za klasifikacijo.

Za namene učenja in sledečega testiranja naučenega modela, celo množico

[1] Tu se le spomnimo, da Evklidska razdalja med točkama $A(a_1, a_2)$ in $B(b_1, b_2)$ predstavlja dolžino daljice med tema dvema točkama $d(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$

podatkov razdelimo v dve množici, učno in testno. Konceptualno je uporaba dveh ločenih množic nujna, saj je to minimalen pogoj, preko katerega zagotovimo osnovno nepristranskost modela. To pomeni, da se bo model učil na enem naboru primerov, končni rezultat pa bo testiran na drugem naboru primerov, za katerega model ne pozna vrednosti razreda. Če bi testiranje potekalo na učni množici, bi dobili popoln rezultat in temu pravimo angl. *overfitting*. Model bi se zdel uspešen, ampak v primeru testiranja na popolnoma novi in neznani množici primerov bi se izkazal za neuspešnega.

Pomembno je tudi, da ves postopek učenja, testiranja in primerjanja napak ponovimo tolikokrat, da lahko zanesljivo potrdimo, da rezultati niso dobljeni po naključju. V ta namen smo uporabili prečno preverjanje, katerega opis sledi v poglavju 2.2.2.

Problem, s katerim se ukvarjamo, je s področja nadzorovanega učenja, saj so nam znane vrednosti razreda, h kateremu pripada vsak primer. Hkrati je problem regresijski, kar napovedujemo oceno od ena do pet. Več o metodah in meritvenih napakah, uporabljenih pri izvedbi diplomske naloge sledi v poglavju 2.2.

2.1.3 Analiza omrežij

Zadnje tematsko področje, preko katerega smo sestavili cenilke, je področje analize omrežij. Osnovna ideja je izkoristiti znanje iz teorije grafov oziroma iz lastnosti grafa, ki ga definiramo kot:

$$G = (V, E). \tag{2.6}$$

Elementa zgornje enačbe (2.6) sta definirana kot:

$$V = \{v_1, v_2, \ldots, v_n\} \tag{2.7}$$

$$E \subseteq \{\{v_i, v_j\} \mid v_j, v_i \in V\}, \tag{2.8}$$

kjer je v enačbi (2.7) n število vozlišč.

Področje temelji na statističnih modelih, s katerimi predstavljamo velika realna omrežja in pri tem izkoriščamo lastnosti grafov, ki jih merimo preko razmerja med vozlišči in povezavami. Primeri takšnih omrežij so:

- biološka omrežja za predstavitev kompleksnih povezav med različnimi beljakovinami in ostalimi kemijskimi komponentami, ki sestavljajo živa telesa;
- nevronska omrežja, ki jih sestavljajo nevronske celice v možganih;
- organizacijska omrežja, ki predstavljajo strukturo določenega podjetja;
- splet s svojo specifično strukturo [11] in
- družabna omrežja za interakcijo med ljudmi, na primer *Facebook*.

Sorazmerno s hitro izmenjavo informacij in tehnološkim napredkom se omrežja izjemno hitro razširjajo. Kot poudarjajo zgornji primeri, je predstavitev podatkov v obliki omrežja naravna in samoumevna, še posebej v primeru velike množice podatkov, ki so v medsebojni interakciji. V tem primeru pričakujemo lažje iskanje vzorcev v podatkih takšne narave. Korak, ki sledi osnovni analizi in iskanju vzorcev, je napovedovanje na podlagi obnašanja posameznih elementov v omrežju.

V nadaljevanju so podani nekateri osnovni pojmi in lastnosti omrežja ter njihov pomen.

Osnovni pojmi in lastnosti

Kot je razvidno iz zgornjih enačb, lahko vsak graf vsebuje dve vrsti povezav, in sicer usmerjene in neusmerjene. Usmerjena povezava poteka v smeri od vozlišča v_1 proti vozlišču v_2 ali obratno; s tem povezave označujejo nesimetrično relacijo med dvema vozliščema. To vpliva na vhodno in izhodno stopnjo vozlišč, kar vpliva na izračun središčnosti in na ostale lastnosti. Temu ustrezno grafu, ki vsebuje usmerjene povezave, pravimo usmerjen graf, medtem ko graf, ki vsebuje neusmerjene povezave imenujejo neusmerjen graf.

Druga klasifikacija grafov na podlagi vrednosti povezav je razdelitev na

uteženе in neuteženе grafe. V prvih ima vsaka povezava različno vrednost oziroma ceno, s katero se uteži pripadajoče vozlišče pri izračunu njegovih lastnosti. Če pa so povezave enakovredne, takšnemu grafu pravimo neuteženen graf.

Premer (angl. *diameter)* defininiramo kot najdaljšo med vsemi najkrajšimi potmi med poljubnima vozliščema v grafu. Pot definiramo kot zaporedje medsebojno povezanih vozlišč, dolžina poti pa je definirana kot število povezav v poti.

Nakopičenost (angl. *clustering coefficient)* je lastnost omrežja oziroma grafa, ki je definirana kot razmerje med povezavami med sosednimi vozlišči in številom sosednih vozlišč:

$$C_i = \frac{2 \cdot e_i}{k_i \cdot (k_i - 1)}, \tag{2.9}$$

kjer je k_i stopnja i-tega vozlišča, e_i pa število povezav med sosednimi vozlišči vozlišča i. Povprečno nakopičenost omrežja izračunamo kot aritmetično sredino vseh posameznih vrednosti C_i:

$$C = \frac{1}{N} \sum_{i}^{N} C_i. \tag{2.10}$$

Središčnosti

Poleg zgoraj naštetih osnovnih lastnosti omrežja poznamo tudi nekaj lastnosti posameznih vozlišč, ki smo jih uporabili v izračunu zadnjega dela, opisanega v poglavju 5.3. Najprej so predstavljene središčnosti (angl. *centrality)* vozlišč.

Stopnja vozlišča (angl. *degree)* neposredno določa osnovno mero, to je stopnjo središčnosti (angl. *degree centrality).* Ta označuje število povezav, ki potekajo v in izven vozlišča. Iz tega sledi, da pri usmerjenih omrežjih

ločimo med vhodno in izhodno stopnjo središčnosti. V primeru neusmerjenih povezav pa je vhodna stopnja vozlišča enaka izhodni. Stopnja vozlišča neposredno označuje število sosednih vozlišč. Preko te mere pa ne dobimo dovolj informacij o pomembnosti vozlišča; tudi v primeru majhne stopnje lahko vozlišče povezuje velike komponente omrežja.

Središčnost vmesnosti (angl. *betweenness centrality)* poleg stopnje vozlišča vpošteva tudi vmesna vozlišča, ki delijo določeno vozlišče od ostalih. Središčnost vmesnosti določa, koliko je vozlišče pomembno, na podlagi tega, kolikokrat ima povezovalno vlogo v najkrajši poti med poljubnimi vozlišči v grafu. Z matematičnim zapisom jo lahko predstavimo kot:

$$C_{BC}(i) = \sum_{i \neq q \neq q} \frac{e_{p,q}(i)}{e_{p,q}}. \tag{2.11}$$

Iz enačbe vidimo, da je BC enaka vsoti razmerij najkrajših poti $e_{p,q}(i)$ med vozliščema p in q, ki gredo skozi vozlišče i, ter vseh najkrajših poti $e_{p,q}$ med p in q.

Bližinsko središčnost (angl. *closeness centrality)* izračunamo s spodnjo enačbo (2.12), kar pomeni, da ima vozlišče i večjo bližinsko središčnost, če do vseh vozlišč j iz grafa, obstaja čim krajša dolžina poti $d_{i,j}$. Cilj je torej priti od vozlišča i do vozlišča j preko čim manjšega števila povezav.

$$C_{CC}(i) = \frac{1}{\sum_{j=1}^{N} d_{i,j}}. \tag{2.12}$$

Središčnost lastnega vektorja (angl. *eigenvector centrality)* je osnova za znan algoritem uvrstitve strani, obrazložen v naslednjem odstavku. Bivstvo središčnosti lastnega vektorja je, da določa pomembnost vozlišča na podlagi pomembnosti njegovih sosedov. V ozadju je ideja, da vsa vozlišča v omrežju

niso enakovredna. Središčnost izračunamo rekurzivno po formuli:

$$C_{EVC}(i) = \frac{1}{\lambda}\sum_{j \neq i} \omega_{i,j} C_{EVC}(j), \tag{2.13}$$

kjer je λ lastna vrednost matrike sosednosti grafa oziroma konstanta, $\omega_{i,j}$ utež povezave med vozliščema i in j ter $C_{EVC}(j)$ EVC za vozlišče j.

Preostali algoritmi

Viri (angl. *authorities)* predstavljajo spletne strani s kakovostno vsebino oziroma vsebovanimi koristnimi informacijami. Kvalitetni viri so spletne strani z visoko vhodno stopnjo vozlišča, prepoznamo pa jih tudi tako, da na njih kaže veliko število kazal. Kazala (angl. *hubs)* so povezave, ki kažejo na vire in igrajo vlogo strokovnjaka, ki priporoča spletne strani. Večja izhodna stopnja vozlišča pomeni večjo vrednost kazala. Kazala in viri (angl. *hubs and authorities)* pod skupnim imenom HITS, predstavljajo algoritem za izračun uteži kazal in virov [10].

Uvrstitev strani (angl. *PageRank)* [26] je izboljšana mera središčnosti lastnega vektorja. *Google* je to mero uporabil ravno za uvrščanje spletne strani, in sicer preko zgodnje verzije njihovih iskalnih algoritmov. PR deluje tako, da strani razvrsti na podlagi tega, kakšne povezave vsebuje določena spletna stran. Bolj konkretno, stran je pomembna toliko, kolikor na njo kažejo ostale pomembne, kakovostne strani. Uvrstitev strani poudari pomembnost povezav do spletne strani, ne pa samo število strani, ki kažejo na njo.

Algoritma HITS in uvrstitev strani se ponavadi izvajata tako dolgo, dokler ne konvergirata do določene vrednosti, ki jo pogosto podamo kot vhodni parameter, s čimer omejimo število iteracij izvajanja algoritmov.

Odkrivanje skupnosti

Ko govorimo o analizi omrežij, se ni mogoče izogniti iskanju skupnosti (angl. *community detection)* [13, 15]. V praktičnem delu pričujoče naloge se

takšne analize ne lotimo zaradi trenutnega obsega in časovne kompleksnosti. Iskanje ali odkrivanje skupnosti lahko sicer izvedemo s pomočjo primitivne tehnike, kot je hierarhično razvrščanje omenjeno v poglavju 2.1.2. Med bolj naprednimi metodami za ta namen sta odkrivanje prekrivanj skupin (angl. *overlapping)* in odkrivanje skupin z bolj naprednimi metodami na podlagi modularnosti grafa. Modularnost nam pove, kako dobro so skupine v grafu ločene med seboj.

Analiza omrežij je obetavno področje ne glede na računsko kompleksnost, s katero se pogosto srečamo ob obdelavi omrežja, še posebej če je le-to iz takšnega razmerja velikosti in kompleksnosti, kot je marsikatero spletno družabno omrežje. Pristopi analize omrežij potencialno ponujajo možnosti ugotavljanja značilnosti za vozlišča in povezave z drugačnega stališča, o katerih ni možno sklepati le na podlagi analize osnovnih atributov.

2.2 Metode strojnega učenja

V nadaljevanju so predstavljene izbrane metode preko katerih smo napovedovali ocene. Algoritmi za učenje in napovedovanje so bili izbrani glede na vrsto in stopnjo zapletenosti: linearna regresija, naključni gozdovi in metoda podpornih vektorjev. Da bi se prepričali o pravilnosti vsakega dobljenega rezultata, smo uporabili tudi metodo prečnega preverjanja. Na koncu poglavja so predstavljene meritvene napake, preko katerih smo rezultate empirično primerjali. Matematične izpeljave presegajo obseg diplomskega dela in zaradi tega v nadaljevanju sledijo le kratki povzetki o idejah in ciljih algoritmov.

2.2.1 Regresijski algoritmi

Sledi predstavitev štirih v praksi uveljavljenih algoritmov različnega tipa. V poglavju 5, kjer so predstavljeni rezultati, je razvidno, kateri od njih se izkaže za relativno uspešnejšega v primerjavi z ostalimi.

Metoda srednje vrednosti

Metoda srednje vrednosti (angl. *mean)* za vrednost napovedi razreda poda srednjo vrednost celega nabora oziroma vektorja razreda. To pomeni, da se najmanj prilagaja posameznim primerom. Če bi to grafično predstavili, bi ugotovili, da model predstavlja nemonotona vodoravna premica, vzporedna x-osi, ki gre hkrati skozi točko srednje vrednosti razrednega nabora na y-osi.

S to metodo v praksi pričakujemo naslabše možne rezultate. Zato jo uporabljamo kot najslabšo napovedovalno metodo in kot osnovno referenco za primerjavo uspeha napovedi ostalih, bolj kompleksnih metod, ki sledijo v nadaljevanju.

Linearna regresija

Najenostavnejši med naštetimi algoritmi je linearna regresija [20, 25]. Kot naslednica metode srednje vrednosti regresija upošteva še odstopanja posameznih vrednostih atributa od srednje vrednosti. Model algoritma ni namreč nič drugega kot linearna funkcija, ki jo definiramo kot:

$$h_\omega = \omega_0 x_0 + \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_m x_m, \tag{2.14}$$

kjer je nabor cenilk enak $(x_0, x_1, x_2, \ldots, x_m)$ in nabor njim ustreznih uteži $(\omega_0, \omega_1, \omega_2, \ldots, \omega_m)$. Definiramo še $x_0 = 1$, da je ω_0 konstantna utež.

Naloga regresije je, da določi uteži, s katerimi se bo regresijska premica najbolje prilagajala učnim podatkom. Poskrbi, da je razlika med napovedano in dejansko vrednost razreda čim manjša. Napovedovalni model je enostaven, vendar se izkaže za manj uporabnega pri nelinearnih podatkih.

Naključni gozdovi

Pri razlagi te metode je potrebno najprej definirati termin odločitveno drevo. V strojnem učenju je odločitveno drevo napovedni model, ki na vsakem novem dodanem nivoju poskuša izločiti primere po določenem atributu na način, da se v vsakem vozlišču drevesa jasno dobi en, čim bolj čisti razred, kateremu pripadajo primeri tega vozlišča.

Poudariti pa je potrebno tudi nekaj pomanjkljivosti odločitvenih dreves, med katerimi izpostavljamo občutljivost na spremembe v učni množici zaradi povečanja napake s povečanjem globine drevesa. Omenjena pomanlkjivost je motivacija za uvedbo metode naključnih gozdov.

Naključni gozdovi [17, 20] je algoritem, zasnovan na pogladi odločitvenih dreves, z namenom izboljševanja njihove napovedovalne moči. Splošen princip njegovega delovanja je, da zgradi več različnih odločitvenih dreves, ki pri napovedi glasujejo v kateri razred naj bo primer uvrščen. Primeru je nato dodeljen razred, ki je dobil večino glasov.

Posebna lastnost metode naključnih gozdov je izbira primerov preko metode stremena (angl. *bootstrap)*, ki ob gradnji gozda za vsako drevo naključno izbere primere. Postopek ponovimo večkrat; ker metodo izvedemo večkrat z različno vhodno množico primerov in rezultate na koncu združimo, temu pravimo angl. *bagging*. Lahko se zgodi, da se primeri ponovijo večkrat pri različni iteraciji ustvarjanja dreves. Cilj je zgraditi čim več različnih dreves, ki so med sabo čim manj korelirana.

Ta algoritem je eden izmed bolj zanesljivih algoritmov v strojnem učenju, saj odpravi pomanjkljivosti odločitvenih dreves, in sicer omenjeno občutljivost na spremembe v učni množici. Težave nastanejo pri interpretaciji modela, saj težko sklepamo o točno izbranih primerih v vsaki iteraciji posameznih dreves gozda.

Metoda podpornih vektorjev

Na splošno velja, da je metoda podpornih vektorjev [14, 20] med najbojšimi algoritmi za klasifikacijo in regresijo zaradi visoke točnosti, ki jo je mogoče doseči. Izhaja iz regresijskih modelov, vendar lahko napove tudi bolj kompleksne nelinearne funkcije.

Imamo dva nabora primerov

$$(x_1, x_2, x_3, \ldots, y_1), \qquad (2.15) \qquad (x_4, x_5, x_6, \ldots, y_2), \qquad (2.16)$$

kjer sta y_1 in y_2 razreda, h katerima pripadajo primeri obeh skupin. Definiramo tudi hiperravnino z enačbo

$$\overrightarrow{w} \cdot \overrightarrow{x} + b = 0, \qquad (2.17)$$

kjer je $\overrightarrow{w}$ normalni vektor ravnine.

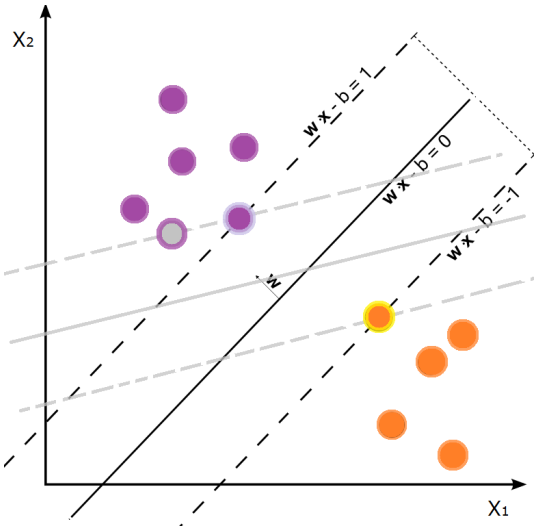
Bivstvo algoritma je razdeliti primere obeh razredov tako, da sta razliki med glavno hiperravnino[2] oziroma robom in najbližjim primerom iz prve oziroma druge skupine, največji. Pri tem se ustvarita dve hiperravnini, po ena na vsaki strani roba:

$$\overrightarrow{w} \cdot \overrightarrow{x} + b = 1, \qquad (2.18) \qquad \overrightarrow{w} \cdot \overrightarrow{x} + b = -1. \qquad (2.19)$$

Preko vektorja $\overrightarrow{w}$ maksimiziramo razdaljo med hiperravninami. Dobljena razdalja je odvisna od posameznih primerov, ki pripadajo najbližji hiperavnini. Te primere imenujemo *podporni vektorji*. Primer takšne razporeditve je prikazan na sliki 2.1.

Hiperravnina se namesto premice uporablja zato, ker se atributi preslikajo v večdimenzionalni prostor, tako da bi postali medsebojno linearno neodvisni. Težava z metodo podpornih vektorjev je v tem kako razložiti dobljen model.

[2]Hiperravnina je posplošitev osnovnega geometrijskega pojma ravnine za poljubno dimenzijo vektorskega prostora.



Slika 2.1: Shema dveh kombinacij hiperravnin, ki razdelita primere na dva razreda, vijolčnega in oranžnega. Podporni vektorji so označeni z različno barvo obrobe kroga [6].

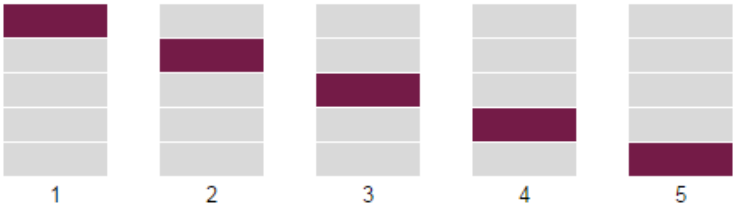
2.2.2 k-kratno prečno preverjanje

Zgoraj omenjeni napovedni algoritmi so zasnovani tako, da se učijo na množicah primerov in testirajo množice primerov, ki jih dobijo kot vhodne parametre. Določanje teh množic je izjemno pomemben del, saj je od tega neposredno odvisna uspešnost algoritma.

Posledično se ne moremo zanašati na rezultate, dobljene iz ene množice, zato je potrebno algoritem nekajkrat preizkusiti na več različnih množicah. Tu se pojavi vprašanje, kako to narediti tako, da bo izkoristek primerov čim večji in da bo upoštevano *de facto* pravilo o ločeni učni ter testni množici.

Ena od možnosti je uvedba *k-kratnega prečnega preverjanja* (angl. *k-fold cross validation)* [19]. Na sliki 2.2 je vizualno predstavljeno, kako se množica razdeli na k enakih naborov. Od teh je $k-1$ naborov učnih, zadnji k-ti nabor (vijolčni) pa testni. V zanki, ki se izvede k-krat, proces ponavljamo tako, da zamenjujemo nabor testne množice. Algoritem se izvaja sproti za vsako od teh k iteracij. Končna vrednost uspešnosti se pogosto izračuna kot aritmetična srednja vrednost rezultatov vseh k iteracij.

Kadar vrstni red primerov ni naključen, se lahko zgodi, da so primeri z istim razredom razporejeni po vrsti, kar povzroča večje napake pri testiranju. S prečnim preverjanjem se tej napaki ponavadi izognimo, ker ne izpustimo primerov pri izvajanju, kar je tudi glavna prednost tega pristopa.



Slika 2.2: 5-kratno prečno preverjanje

2.2.3 Ocenjevanje uspešnosti modelov

Da bi lahko primerjali modele med seboj, smo primerjali vrednosti nekaterih meritvenih napak z različnimi lastnostmi. Vsaka vrednost meritvene napake je izračunana za vsako kombinacijo v kateri model nastopa. Vse te različne kombinacije nato primerjamo preko primerjave vrednosti dobljenih napak.

Najprej smo definirali osnovno napako e_i kot:

$$e_i = y_i - f_i, \tag{2.20}$$

kjer je f_i napovedana vrednost razreda, y_i pa dejanska vrednost razreda.

Povprečna vrednost vrednosti učnega razreda je definirana kot:

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.21}$$

V skladu z regresijsko naravo problema v nadaljevanju sledijo tri izbrane mere uspešnosti, s katerimi smo ocenjevali uspešnost napovedovanja.

MAE

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|e_t| \tag{2.22}$$

Iz definicije (2.22) vidimo, da je povprečna absolutna napaka dobljena kot povprečna vrednost vseh posameznih absolutnih vrednosti napake, izračunane za vsak posamezni primer. To pomeni, da MAE pove, za koliko točk v povprečju dejanska vrednost odstopa od napovedane vrednosti.

Hiba te meritve je v tem, da dodeli isto utež oziroma kazen vsem napovedanim vrednostim, ne glede na njihova odstopanja.

RMSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{t=1}^{n}e_t^2} \tag{2.23}$$

Enačba (2.23) prikazuje koren povprečne kvadratne napake. Prednost meritve je, da zaradi kvadriranja osnovne napake e_i, večja odstopanja v napovedi f_i dobijo višjo kazen. Vse osnovne napake torej ne glede na njihovo vrednost niso enakovredne. Koren nam omogoča, da je mera v istem razmerju kot vhodni podatki. RMSE je vedno večji ali enak napaki MAE [12].

R^2

$$R^2 = \frac{pojasnena\ varianca}{skupna\ varianca} = \frac{\sum_{i=1}^{n} e_i^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}, \tag{2.24}$$

kjer so e_i, y_i in $\overline{y}$ definirani v enačbah (2.20) in (2.21). Koeficient determinacije določa razmerje med pojasnjeno in skupno varianco. To pomeni, da

lahko model z vrednostjo R^2 odstotkov določi točno toliko odstotkov variance v ocenjenem atributu. Zaloga vrednosti napake je interval [0, 1], kjer 1 označuje popolno prilagajanje podatkom.

Pomanjkljivosti osnovne verzije koeficienta so te, da raste monotono z večanjem števila spremenljivk oziroma atributov, kar lahko napačno kaže na izboljšano napoved.

Poglavje 3

Programska orodja

Veliko časa pri pripravi diplomske naloge je bilo posvečenega knjižnicam za podatkovno rudarjenje in strojno učenje, opisanim v podpoglavjih 3.1.1 in 3.1.2, saj je bil eden izmed ciljev naloge tudi seznanitev z novim programskim okoljem. Vsa našteta in opisana orodja so odprtokodna. Dosedanje pozitivne izkušnje z izbranimi orodji so bile še dodaten razlog za izbiro le-teh. Sledi kratek povzetek njihovih lastnosti in funkcionalnosti.

3.1 Python

Skriptni programski jezik Python je bil prva izbira pri opravljanju praktičnega dela, saj je sintaktično enostaven in intuitiven za uporabo. Ravno zato je priljubljen tudi med osebami, ki niso vešči programerji. Njegova največja prednost je velika izbira knjižnic za delo s podatki, ki dobro dopolnjujejo osnovne funkcionalnosti tega jezika [24]. Zaradi tega je ta jezik izjemno popularen na področju podatkovnega rudarjenja in strojnega učenja.

3.1.1 pandas

`pandas` [3] je knjižnica, ki omogoča optimalno zapisovanje podatkov v podatkovnih strukturah, kar je več kot priporočeno pri delu z velikim številom primerov in atributov. Napisana za Python, da jeziku dodatno moč za hitro

in enostavno manipulacijo s podatki.

V knjižnici je osnovna podatkovna struktura *DataFrame*, ki igra vlogo dvodimenzionalne tabele, kjer je možno poimenovati vrstice in stolpce, pri čemer dobimo slovar. Prednost te strukture proti ostalimi je, da lahko vsebuje več različnih tipov podatkov, o katerih lahko hrani tudi dodatne informacije o tipu in velikosti podatka. Podatki znotraj strukture so enako poravnani, kar je pomembna lastnost pri branju in zapisovanju datotek. Dodatna prednost je tudi to, da omogoča preprosto in kratko pogojno iskanje po podatkih znotraj same strukture. Poleg tega pretvorba iz osnovnih slovarjev in nizov ter obratno ni zahtevna.

V pričujoči nalogi je podatkovna struktura uporabljena tako, da so atributi oziroma cenilke predstavljeni po stolpcih, vsak primer pa predstavlja nepoimenovana vrstica. V takšni obliki je iz podatkovne strukture zelo enostavno brati, v njo pisati ali obdelovati stolpce in vrstice. To omogoča učinkovitejši pregled rezultatov in enostavno sortiranje primerov po željenih pogojih. Možen je izvoz strukture v LaTeX obliki. Za osnovni zapis enodimenzionalnih nizov v okviru knjižnice pandas smo kot pomožno knjižnico uporabili `NumPy` [2].

3.1.2 scikit-learn

`Scikit-learn` [4, 27] je knjižnica za podatkovno rudarjenje in strojno učenje, napisana za jezik Python. Vsebuje funkcionalnosti, s katerimi podatke analiziramo od začetka do konca preko normalizacije, učnih in testnih modelov ter preverjanja s pomočjo meritvenih napak. Splošna razširjenost uporabe je predvsem zaradi enostavne izvedbe velikega števila modelov, ki jih knjižnica vsebuje. Na ta način obdržimo neposredni vpogled v procese priprave podatkov, učenja in testiranja. To pomeni, da se lahko osredotočimo na odločanje o izbiri lastnosti, modelov in parametrov, ne da bi porabili preveč časa za dejansko programiranje algoritmov.

Za namen pričujoče naloge smo uporabili različne module in razrede. Po-

membnejši so prikazani v spodnji tabeli.

Modul/razred	Uporaba
`preprocessing.normalize`	Normalizacija vrednosti atributov
`cross_validation`	k-kratno prečno preverjanje
`linear_model.LinearRegression`	Linearna regresija
`ensemble.RandomForestRegressor`	Naključni gozdovi
`svm.LinearSVR`	Metoda podpornih vektorjev

`mean_absolute_error`
`mean_squared_error`
`r2_score`
	Meritvene napake

`TfidfVectorizer`	Izdelava vreče besed Izračun frekvence besed

3.2 Snap.py

Knjižnica SNAP [5, 21] omogoča funkcije za delo s podatkovnimi omrežji. Za namene naloge in v skladu z izbranim programskim jezikom smo uporabljali različico za programski jezik Python, poimenovano kot `Snap.py`.

Delo s Snap.py je enostavno, sintaksa pa zagotavlja natančni pregled operacij nad omrežij. Knjižnico podpira tudi dobro sestavljena dokumentacija. Omogoča začetno gradnjo grafov in vsebuje izjemno širok nabor različnih funkcij za iskanje lastnosti omrežja ter iskanje skupnosti. S pomočjo te knjižnice smo pridobili lastnosti posameznega omrežja, središčnosti in ostalih lastnosti posameznega vozlišča, omenjenih v poglavju 2.1.3.

3.3 Gephi

Grafični vmesnik Gephi [1, 8] zagotavlja vizualno delo z omrežji in je neodvisen od programskega jezika. Uporabili smo ga kot dopolnitev knjižnice Snap.py, in sicer za grafični izris omrežja. Kot vhodni podatek smo podali tekstovno datoteko, ki je vsebovala omrežje v standardnem zapisu povezav,

to je z začetnim in končnim vozliščem.

Največja prednost orodja je velika izbira možnih predstavitev omrežij in omogočanje vizualnega izpostavljanja vozlišč in povezav ter sprememinjanje oznak, barv in velikosti elementov. Posebej zanimive so različne ponujene animacije premikanja omrežja z namenom preoblikovanja vidne strukture. Poleg tega program omogoča izračun lastnosti omrežja ter njihov izvoz v poljubni obliki za nadaljno analizo.

Poglavje 4

Podatki in določanje cenilk

V nadaljevanju sledi opis izbrane množice primerov, na kateri smo testirali modele. Ta del je pokrit s strani podatkovnega rudarjenja, saj je bilo potrebno izluščiti zaželjene podatke. Na potek praktičnega dela je imela precejšnji vpliv oblika množice primerov.

4.1 Množica primerov

Vir podatkov je spletna zbirka podatkovnih množic Univerze Stanford, in sicer zbirka ocen filmov v spletni trgovini *Amazon* [7]. Nabor vključuje podatke od augusta 1997 do oktobra 2012.

Skupno število primerov je nekaj manj kot osem milijonov. Število posameznih uporabnikov znaša okrog 890 000, število izdelkov oziroma filmov, ki so bili komentirani, pa je okrog 250 000. Najmanj 16 000 uporabnikov je ocenilo in komentiralo več kot 50 različnih filmov. Zaradi praktičnih razlogov ter prostorskih in računskih omejitev, smo za nadaljnjo analizo izbrali ocene filmov iz leta 2010, ki vsebuje približno 750 000 primerov.

4.2 Oblika posameznega primera

Posamezen primer vsebuje podatke za posamezno filmu dodeljeno oceno s strani uporabnika. Podrobnosti so podane v tabeli 4.1, iz katere je razvidno, da nam je takšna struktura podatkov narekovala potek dela po fazah, ki so podrobno opisane kasneje.

Atribut (orig. v angl.)	Pomen atributa
productId	evidenčno ime izdelka
userId	evidenčna številka uporabnika
profileName	ime uporabniškega računa
helpfulness	razmerje, ki pove, koliko ostalih uporabnikov je glasovalo, da jim je ta ocena koristila
score	številčna ocena v vrednosti od "1"do "5"
time	čas oddaje ocene
summary	naslov komentarja
text	besedilo komentarja

Tabela 4.1: Oblika posameznega primera v naboru podatkov

Lastnosti lahko neformalno razdelimo v skupine, in sicer v splošne podatke o uporabniku in izdelku, oceno, besedilne podatke o oceni, časovni podatek ter odziv ostalih uporabnikov. Kot razred, ki smo ga napovedovali, smo izbrali atribut *score*, ki ga v nadaljevanju naslavljamo kot *ocena*.

4.3 Koncept določanja cenilk

Glavni cilj naloge je medsebojno primerjati poljubno sestavljanje atributov oziroma cenilk, preko katerih bi čim bolj pravilno napovedali številčne ocene filmov. Iz množice, predstavljene v prejšnjem podpoglavju, smo začeli z najbolj enostavnimi oziroma primitivnimi cenilkami, ki jih je bilo moč ugotoviti preko osnovne obdelave atributov surovih primerov. Najbolj nas je

zanimalo, kako se bosta izkazala atribut koristnosti in časovni atribut.

Nato smo se lotili osnovne analize besedila komentarja iz vsakega primera in iz tega sestavili množico cenilk. Tako smo dobili vpogled v povezanost komentarjev z ustreznimi ocenami.

Na koncu pa je sledilo bolj kompleksno modeliranje cenilk. Izkoristili smo znanje iz analize omrežij, in sicer cenilke predstavljajo lasnosti vozlišč in povezav dobljenega grafa na podlagi primerov. V ta namen smo izkoristili splošne podatke o uporabnikih in izdelkih.

4.4 Začetno testiranje

Na začetku testiranja željenega koncepta in z željo, da bi imeli čim več primerov v učni in v testni množici, smo uporabili skoraj vseh 750 000 primerov iz leta 2010, pri katerih jih je okoli 515 000 vsebovalo pozitivni oziroma neničelni atribut koristnosti. V tabeli 4.2 so prikazani rezultati preizkusa metode srednje vrednosti. Zaradi preprostosti modela je predstavljena samo meritev RMSE, dobljena kot povprečna vrednost 10-kratnega prečnega preverjanja.

Model ima določene pomankljivosti, predvsem pri porazdelitvi vrednosti razredov, ki pa smo jih odpravili v nadaljevanju. Iz tega modela tako ne moremo sklepati ničesar razen tega, da velikost nabora primerov (tako učnih kot testnih) vpliva na izboljšanje napovedovanja, kar vidimo iz nižje vrednosti meritve napak.

RMSE	Število primerov	RMSE	Število primerov	Velikost učne množice
1,411	515 000	1,292	750 000	80 %
1,413	515 000	1,294	750 000	60 %

Tabela 4.2: Metoda srednje vrednosti - začetno testiranje

4.5 Priprava testnih atributov

Celotni nabor razredov v podatkovni množici ni bil enakomerno porazdeljen. Kot primer je navedeno naslednje razmerje med dvema ocenama, in sicer

$$\text{"2"} : \text{"5"} = 1 : 10.$$

Da bi to pomanljkljivost odpravili, saj bi negativno vplivala na učni model, in hkrati dobiti čim bolj natančne rezultate, smo generirali naključne podatkovne množice z enako porazdeljenimi razredi.

Za namen osnovnega testiranja smo uporabljali dve različni velikosti podatkovnih množic. Prva sestoji iz množic s 1000 primeri, od katerih posamezna vključuje 200 primerov iz vsakega razreda. Druga pa vsebuje 200 000 primerov oziroma 40 000 primerov iz enega razreda. Poleg omenjenih smo uporabljali še pet drugih velikosti množic, in sicer z 10 000, 25 000, 50 000, 100 000 in 150 000 primeri.

Vrstni red primerov v množicah je nekajkrat premešan, s čimer smo dosegli večjo naključnost in se izognili pretiranemu prilaganju enemu razredu.

Poglavje 5

Rezultati in diskusija

Pričujoče poglavje se navezuje na začetno idejo, opisano v poglavju 4.3. Tu so namreč predstavljeni številčni rezultati praktičnega dela, ki jim sledi diskusija. Med seboj so primerjane vse tri faze dela, prav tako pa so navedene možnosti izboljšav znotraj posameznih faz, v kolikor je bilo mogoče testirati več možnih načinov sestavljanja cenilk. Rezultati vsake faze so sicer neposredno neodvisni od preostalih dveh, vendar je bilo pri zgradbi modelov potrebno omogočiti skupno osnovo za primerjavo.

5.1 Prva faza: osnovni atributi

V začetni prvi fazi smo na podlagi metode srednje vrednosti najprej določili najslabše pričakovane napovedane vrednosti. Temu so sledili poskusi primerjanj različnih kombinacij osnovnih atributov, nato pa še testiranje s pomočjo kompleksnejših metod poglavja 2.2.1.

5.1.1 Napoved srednje vrednosti

Da bi pridobili izhodiščno točko primerjave, je bilo potrebno najprej preizkusiti model z najslabšo možno pričakovano napovedjo, s katero bi lahko neposredno primerjali napovedi ostalih testiranih metod. Pričakovali bi, da so druge napovedi uspešnejše od najslabše spodnje meje rezultatov. Kot naj-

preprostejši model smo izbrali metodo srednje vrednosti (angl. mean), ki kot napoved razreda poda srednjo vrednost elementov razrednega nabora. To pomeni, da se najmanj prilagaja posameznim primerom. Povprečni rezultati te metode pri 10-kratnem prečnem preverjanju so razvidni s tabele 5.1.

MAE	RMSE	R^2	Število primerov	Velikost učne množice
1,198	**1,411**	**0**	**10 000**	**80 %**
1,201	1,413	0	10 000	60 %
1,200	1,414	0	200 000	60 %
1,201	1,415	0	200 000	80 %
1,209	1,421	-0,003	1 000	60 %
1,211	1,423	-0,002	1 000	80 %

Tabela 5.1: Metoda srednje vrednosti. Izhodiščna točka za primerjavo z naslednjimi modeli. Urejeno naraščajoče po RMSE.

S tabele lahko sklepamo, da so rezultati precej slabi, predvsem zaradi negativne vrednosti R^2 koeficienta, ki v najboljšem primeru dobi vrednost 1.0. Negativna vrednost nam pove, da model v povprečju ne uspe narediti uspešne napovedi pričakovane vrednosti. V primeru največjega števila primerov (to je 200 000) metoda ni dosegla relativno najboljšega rezultata. Razlika med najboljšim in najslabšim RMSE znaša 1 %, kar je relativno nizka razlika in se sklada z našimi pričakovanji. Imamo namreč samo en atribut, ki se ne spreminja. V povprečju vsaka napoved odstopa za približno 1,2 točke od povprečja v testni množici.

5.1.2 Filtriranje osnovnih atributov

Pri nadaljni izbiri cenilk je bilo potrebno izluščiti osnovne atribute, ki so na razpolago. Smiselna in potencialno obetavna izbira sta bila osnovna atributa *koristnost* (angl. *helpfulness*) in *časovna značka* (angl. *time*). Kot razred je bil zopet izbran atribut *ocena* (angl. *score*). Atributa *productId* in *userId* v tej fazi nista bila izkoriščena, in sicer zaradi možnosti smiselne

implementacije v naslednji fazi izbire cenilk.

Preko časovne značke smo pridobili cenilke, prikazane v tabeli 5.2. Podatek o točni uri, minuti in sekundi ni bil na voljo.

Cenilka	Zaloga vrednosti	Izpeljana cenilka	Zaloga vrednosti
Dan v tednu	[0, 6]	Dan v tednu	[0, 6]
		Vikend	{0, 1}
Dan v mesecu	[1, 31]	Dan v mesecu	[1, 31]
Dan v letu	[1, 366]	Dan v letu	[1, 366]
		Praznik	{0, 1}
Mesec	[1, 12]	Mesec	[1, 12]
		Letni kvartal	[1, 4]
		Letni čas	[1, 4]
		Polovica leta	{1, 2}

Tabela 5.2: Časovni atributi

V kombinaciji z ostalimi atributi smo na podlagi izpeljanosti ali pomenskosti sestavili štiri skupine. Neformalna povezanost cenilk je naslednja:

1. {vikend, praznik}
2. {koristnost}
3. {mesec}
4. {dan v tednu, dan v mesecu, dan v letu}
5. {letni kvartal, letni čas, polovica leta}

Nabor atributov skupaj šteje deset cenilk.

Zgornja razporeditev po skupinah nam je omogočila hitrejše preverjanje več možnih kombinacij. Naključno je bilo izbranih 1000 primerov, testiranje pa je bilo izvedeno z 10-kratnim prečnim preverjanjem. Sprva smo za napovedovanje izbrali najenostavnejšega med tremi predlaganimi algoritmi, to je linearna regresija. Z njo je bilo testiranih 25 kombinacij zgornjih cenilk. Rezultati, pridobljeni z uporabo tega algoritma, so se izkazali kot slabši v primerjavi z rezultati metode srednje vrednosti, in sicer je koristnost dobila najboljšo mero napake RMSE, ki znaša 1,417. Ker nam algoritem linearne

regresije poda tudi neveljavno vrednost napake R^2, se za primerjavo ne moremo zanašati na negativne vrednosti tega koeficienta. Zato je bilo potrebno testirati še z metodo, ki se bolje obnese v praksi, primer česar je metoda naključnih gozdov.

Rezultati tega testa, in sicer na 1000 primerih, so prikazani v tabeli 5.3. Napovedovalna točnost je malo višja kot tista pri linearni regresiji. V primeru 100 zgrajenih dreves, dobi kombinacija vseh 10 cenilk precej boljšo oceno in ni več najslabša kombinacija. Vendar pa v primeru, da je generiranih le 10 ali 50 dreves, ostaja v primerjavi z drugimi najslabša cenilka.

Iz rezultatov lahko sklepamo, da atributi, ki izhajajo iz časovne značke, ne predstavljajo dobre izbire cenilk za napovedovanje ocene filma in jih zato ne uporabljamo pri nadaljnem testiranju modela. Cenilka z najnižjo napako korena kvadratne vrednosti je cenilka *koristnost*. Od vseh atributov, ima ta cenilka v svojem naboru najbolj različne vrednosti, kar je v soglasju z visoko uspešnostjo napovedi.

Za primerjavo rezultatov pri precej večjem številu primerov (to je 200 000), je podana tabela 5.5. Opazimo lahko, da so rezultati s 60 % učno množico blizu tistim z 80 %. Velik skok v natančnosti se pojavi pri kombinaciji vseh 10 cenilk, in sicer od 0,006 na 0,278 pri koeficientu R^2. To pomeni, da je pojasnjene več variance ali natančneje, da se 28 % variance v podatkih da pojasniti preko konkretnega modela metode naključnih gozdov.

Za lažjo primerjavo z drugo fazo smo napovedi ocen z množice primerov velikosti 200 000 preizkusili tudi z metodo podpornih vektorjev. Kljub nižji uspešnosti so rezultati podani v tabeli 5.6. Glede na nižji uspeh algoritmov linearne regresije in metode podpornih vektorjev lahko sklepamo, da so cenilke v nelinearnem odnosu. Primer takšne relacije je podan na sliki 5.1.

Primerjava rezultatov pokaže, da ne moremo trditi, da je uporaba osnovnih atributov boljša od napovedovanja z metodo srednje vrednosti. Sklepamo lahko, da je dosežena uspešnost v določenih primerih odvisna predvsem od regresijskega algoritma in njegove sposobnosti interpretirati stopnjo medsebojne odvisnosti atributov, bodisi če je ta linearna ali ne.

Kombinacija	MAE	RMSE	R^2	Število dreves
	1,076	**1,291**	**0,166**	**100**
{koristnost}	1,076	1,292	0,164	50
	1,079	1,296	0,159	10
	1,111	1,339	0,101	100
{koristnost, mesec}	1,115	1,344	0,095	50
	1,125	1,357	0,077	10
	1,129	1,371	0,058	100
{vikend, praznik, koristnost, mesec}	1,135	1,375	0.052	50
	1,152	1,398	0,012	10
{vseh deset cenilk}	1,150	1,408	0,006	100
{vikend, praznik}				
{vikend}				
{praznik}				
cenilke po dnevu (glej tabelo 5.4)				
cenilke po letu (glej tabelo 5.4)	1,209	1,418	-0,007	100
{mesec}				
{letni čas, mesec}				
{letni čas, vikend, praznik}				
{letni čas, kvartal, 1/2 leta, mesec}				
	1,157	1,419	-0,009	50
{vseh deset cenilk}	1,176	1,447	-0,049	10

Tabela 5.3: RF pri prvi fazi izveden na 1000 primerov pri 80 % velikosti učne množice. Cenilka koristnost se izkaže kot najuspešnejša. Spodnja polovica tabele prikazuje slabo ocenjene kombinacije. Urejeno naraščajoče po RMSE.

cenilke po dnevu	{dan v tednu, v mesecu, v letu}
	{dan v tednu, dan v mesecu}
	{dan v tednu, dan v letu}
	{dan v mesecu, dan v letu}
	{dan v tednu}
	{dan v mesecu}
	{dan v letu}
cenilke po letu	{letni kvartal, letni čas, 1/2 leta}
	{letni kvartal, letni čas}
	{letni kvartal, 1/2 leta}
	{letni čas, 1/2 leta}
	{letni kvartal}
	{letni čas}
	{1/2 leta}

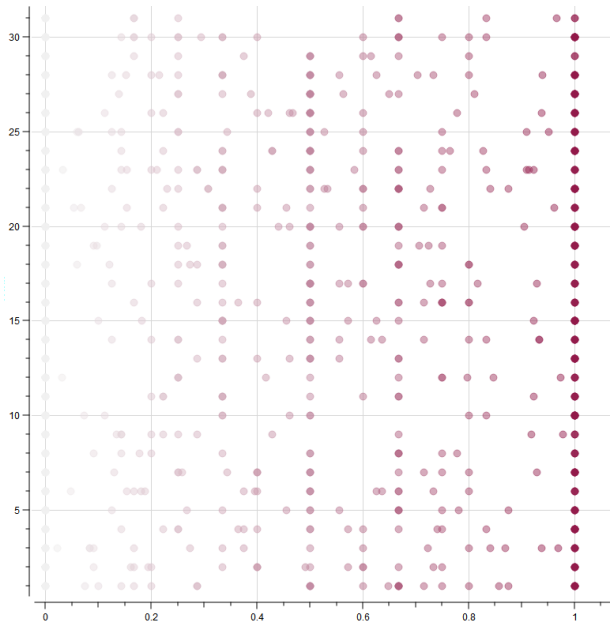
Tabela 5.4: Del kombinacij cenilk, pridobljenih preko časovnega atributa.

Kombinacija	MAE	RMSE	R^2	Velikost učne množice
{vseh deset cenilk}	0,947	1,202	0,278	80 %
	0,958	1,212	0,264	60 %
{vikend, praznik, koristnost, mesec}	1,016	1,236	0,236	80 %
	1,019	1,239	0,231	60 %
{koristnost, mesec}	1,025	1,241	0,231	80 %
	1,027	1,243	0,227	60 %
{koristnost}	1,041	1,247	0,222	80 %
	1,040	1,247	0,221	60 %

Tabela 5.5: RF pri prvi fazi izveden na 200 000 primerov z generiranjem 100 dreves. Prikazani so le najbolj uspešni atributi. Urejeno naraščajoče po RMSE.

Kombinacija	MAE	RMSE	R^2
{vikend, praznik, koristnost, mesec} {vseh 10 cenilk} {koristnost} {koristnost, mesec}	1,200	1,414	0

Tabela 5.6: SVM pri prvi fazi izvedena na 200 000 primerih pri 80 % učni množici. Algoritem se izkaže kot neuspešen pri učenju v prvi fazi.



Slika 5.1: Nelinearna odvisnost med cenilkami. Na x-osi je predstavljena koristnost, na y-osi dan v mesecu

5.2 Druga faza: analiza komentarjev

V drugi fazi je izbor cenilk potekal na podlagi opisne ocene, ki je v vsakem primeru spremljal številčno oceno. To smo izvedli tako, da smo najprej sestavili vrečo besed (angl. *bag of words*), ki je vsebovala vse unikatne besede iz vseh opisnih ocen, ki smo jih testirali. Nato smo za vsako besedo iz vreče izračunali inverzno frekvenco *tfidf*. Kot je zapisano v poglavju 2.1.1. to pomeni, da smo upoštevali frekvenco besed znotraj:

- komentarja posameznega primera in
- celotne množice primerov.

Pri tem smo uporabili razred *TfidfVectorizer* iz programske knjižnice *scikit-learn*, ki je kot rezultat vrnil matriko inverznih frekvenc. Ta matrika je nato predstavljala naš nov nabor atributov.

V okviru tega enostavnega modela smo z namenom izboljšave filtirali in izločili besede brez posebnega pomena, kot so vezniki in predlogi (angl. *stop words*). Vektor teh besed smo empirično določili tako, da smo izpustili besede s frekvenco (znotraj množice primerov) višjo kot 0,3.

Primer vsebine enega izmed vektorjev je sledeč: *{'and', 'all', 'is', 'it', 'one', 'as', 'are', 'have', 'in', 'if', 'from', 'for', 'movie', 'to', 'you', 'was', 'be', 'that', 'but', 'br', 'not', 'with', 'on', 'this', 'of', 'the', 'my'}*. Če bi prag znižali na 0,25, bi se v vektorju pojavile še besede *{'good', 'great', 'like'}*, ki jih intuitivno ne bi bilo smiselno ignorirati, saj potencialno opisujejo lastnosti filma.

Najprej smo z 10-kratnim prečnim preverjanjem testirali 1000 primerov. Kot kaže tabela 5.7, kjer so povzeti rezultati, je bilo pri tem število atributov oziroma različnih besed 14 992. Zaradi časovne kompleksnosti izvajanja metode naključnih gozdov smo generirali le 10 in 50 dreves. Primerjava algoritmov pokaže, da se je pri tem kot najslabši algoritem izkazal metoda naključnih gozdov, saj ta na manjšem naboru primerov ne more dobro izvesti naključne izbire primerov. V primerjavi z njim pa sta se dobro izkazala algoritma metode podpornih vektorjev in linearne regresije, ker omogočata,

da se besede medsebojno linearno razdelijo, poleg tega pa pri linearni metodi podpornih vektorjev preslikovanje v več dimenzionalnem prostoru ne igra nobene vloge.

Metoda	MAE	RMSE	R^2	Število dreves	Velikost učne množice
SVM	1,002	1,185	0,290	/	80 %
LR	1,002	1,206	0,264		80 %
LR	1,011	1,211	0,272		60 %
SVM	1,029	1,214	0,269		60 %
RF	1,041	1,263	0,193	50	80 %
	1,086	1,307	0,153		60 %
	1,089	1,326	0,108	10	80 %
	1,117	1,363	0,079		60 %

Tabela 5.7: Rezultati za 14 992 cenilk, pridobljeni s testiranjem 1000 primerov. Urejeni naraščajoče po meri RMSE.

Nadalje smo testirali tudi, kakšne rezultate bi dobili na istem naboru primerov, če bi zmanjšali nabor cenilk in upoštevali samo določeno število besed iz vreče, ki imajo najvišjo frekvenco *tf*. Te rezultate prikazuje tabela 5.8.

Sklepamo lahko, da bi s trikrat manjšo množico cenilk dobili rezultate relativno podobne prejšnjim, ki se medsebojno razlikujejo za 0,5 %. Če je v naboru 5000 cenilk, so dobljeni rezultati smiselni, saj je koeficient R^2 nenegativen. Če pa je nabor cenilk manjši, na primer 1000 cenilk, lahko poleg dobrih dobimo tudi slabe napovedi.

Na podlagi rezultatov, prikazanih v tabeli 5.8, dobljeni z metodo podpornih vektorjev, smo za testiranje celotne množice primerov izbrali metodo podpornih vektorjev, saj je ta algoritem dal najboljši rezultat. Poleg tega je dodatna prednost tega algoritma v primerjavi z linearno regresijo in naključnimi gozdovi hitrejše izvajanje. Večji nabor primerov, to je 200 000

Metoda	Število cenilk	MAE	RMSE	R^2	Število dreves	Velikost učne množice
SVM	5000	0,992	1,180	0,295	/	80 %
		1,015	1,205	0,280		60 %
	1000	1,001	1,211	0,256		80 %
		1,015	1,228	0,252		60 %
LR	5000	1,016	1,232	0,247	/	60 %
SVM	500	1,009	1,236	0,225		80 %
LR	5000	1,024	1,245	0,214		80 %
SVM	500	1,035	1,258	0,215		60 %
RF	500	1,048	1,270	0,183	50	80 %
	5000	1,045	1,270	0,183		
LR	10	1,228	1,429	-0,012	/	60 %
SVM		1,237	1,454	-0,048		
RF	10	1,241	1,464	-0,083	50	80 %
		1,262	1,484	-0,091		60 %
		1,258	1,491	-0,125	10	80 %
		1,274	1,506	-0,124		60 %
LR	1000	1,390	1,744	-0,514	/	60 %
	500	1,462	1,825	-0,710		80 %
	1000	1,910	2,410	-1,979		80 %
	500	2,177	2,787	-2,889		60 %

Tabela 5.8: Rezultati za cenilke, dobljeni s testiranjem 1000 primerov na manjši množici cenilk. Prikazanih je deset najboljših in najslabših rezultatov. Urejeni naraščajoče po RMSE.

primerov, smo testirali z metodo podpornih vektorjev [18], kar prikazuje tabela 5.9.

Opazimo lahko, da se s povečanjem števila primerov iz 1000 na 200 000 precej poveča tudi množica cenilk, in sicer približno za devetkrat. Ne glede na število primerov je deset besed premalo, da bi dobili kakršnekoli smiselne rezultate. Z maksimalnim naborom cenilk dobimo 64% pokritost variance in temu ustrezna vrednost napake RMSE znaša 0,85, kar je v primerjavi z dosedanjimi rezultati najboljša ocena (natančneje, v primerjavi z rezultati v tabeli 5.6, kjer napaka RMSE znaša 1,414, in sicer tudi za najboljše kombinacije cenilk). To potrjuje tudi napaka MAE, ki je skoraj dvakrat boljša, kar pomeni, da napovedana vrednost v povprečju odstopa za 0,57 točk od dejanske vrednosti razreda.

Obnašanje metod smo testirali še v primeru nižje meje frekvence znotraj korpusa dokumentov. To pomeni, da nismo upoštevali besed iz zgoraj omenjenega vektorja, med katerimi so *good*, *great*, *like*. Presenetljivo razlike v rezultatih niso velike, saj odstopanje od rezultatov v tabeli 5.9 pri istih testnih pogojih znaša samo 0,01.

Več o semantinčni povezavi med opisno oceno in ustrezno številčno oceno na tem nivoju analize ne moremo sklepati, saj enostavnost frekvenčne analize ne omogoča podrobnega vpogleda v medsebojne interakcije besed. Kljub temu lahko zaključimo, da dobimo s preprosto frekvenčno analizo besedila relativno dobre rezultate, ki so prav tako precej boljši od rezultatov prve faze analize. V tem primeru je bil nabor cenilk ožji in z manjšo zalogo vrednosti za vsako cenilko, izjema je bila le cenilka koristnosti.

Število cenilk	MAE	RMSE	R^2
133 368	**0,572**	**0,849**	**0,640**
10 000	0,699	0,943	0,556
5 000	0,739	0,966	0,534
1 000	0,828	1,040	0,460
500	0,879	1,091	0,406
100	1,055	1,278	0,185
10	1,201	1,415	-0,000

Tabela 5.9: SVM na 200 000 primerih pri 80 % učni množici. S povečanjem korpusa besed, se rezultati vidno izboljšujejo. Urejeni naraščajoče po RMSE.

5.3 Tretja faza: atributi omrežja

Da bi nadaljevali s primerjavo izbire cenilk pri napovedovalnih modelih, smo v naslednjem koraku uporabili nekaj osnovnih znanj iz analize omrežij, omenjenih v poglavju 2.1.3. Ta znanja so nam omogočila sestaviti nove cenilke.

5.3.1 Na osnovi primerov

V skladu z določenim razredom v zgoraj omenjenih fazah smo tudi v zadnji fazi kot razred privzeli atribut *ocena*. Ker je vsaka ocena predstavljena z natanko enim primerom, smo za vozlišči v nadalje dobljenem grafu določili prav posamezne primere. Torej, ena ocena oziroma primer predstavlja eno vozlišče v grafu. Kar se tiče povezav, smo se odločili privzeti dejstvo, da je več različnih primerov ocenil isti uporabnik. To pomeni, da sta vozlišči povezani v primeru, če se nanašata na istega uporabnika.

Naslednji korak je bilo generiranje omrežij, in sicer z upoštevanjem uporabe istih primerov tekom celotnega testiranja v vseh treh fazah. S tem namenom smo preko programske knjižnice *Snap.py* ustvarili omrežja z istim naborom primerov, prikazanih v tabeli 5.10. Število vozlišč je nekoliko manjše od števila primerov, ker v omrežjih nismo upoštevali vozlišč z ničelno

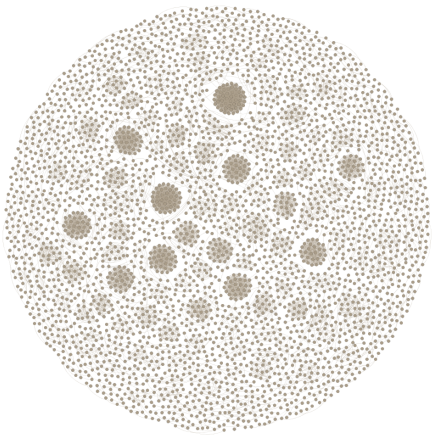
stopnjo, to je vozlišč brez sosednjih vozlišč. Vozlišča brez povezav pomenijo, da je uporabnik podal le eno oceno, kar pa nam zaradi načina zasnove omrežij, ni koristilo.

Število primerov	Število vozlišč	Število povezav	Povprečna stopnja vozlišča	C	D
1 000	134	149	2,2	0,388	1
10 000	3 724	11 278	6,1	0,608	1
25 000	12 710	72 877	11,5	0,670	1
50 000	31 600	275 773	17,5	0,731	1
100 000	74 667	1 128 398	30,2	0,799	1
150 000	120 523	2 550 665	42,3	0,841	1
200 000	167 607	4 469 745	53,3	0,868	1

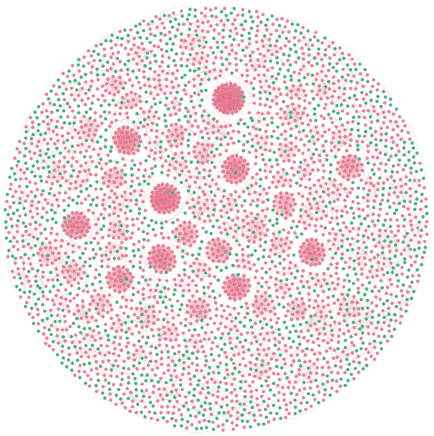
Tabela 5.10: Nekatere lastnosti dobljenih omrežij v eni iteraciji. C označuje nakopičenost, D pomeni premer omrežja.

Iz tabele lahko opazimo tudi, da se število povezav z naraščanjem števila primerov značilno povečuje, saj lahko veliko več primerov izhaja od istih uporabnikov. V največji množici primerov uporabnik v povprečju poda nekaj več kot 50 ocen. Izračunan približek premera povezanih komponent omrežja znaša ena, kar kaže na veliko število izoliranih vozlišč. Nakopičenost raste s povečevanjem števila vozlišč in povezav, kar je ravno obratno, kot pri generiranju naključnega grafa z istim številom elementov. Za primerjavo naj navedemo, da nakopičenost pri naključnem grafu za 1000 primerov znaša 0,019, za 200 000 primerov pa 0,0003, kar je precej manj kot pri realnemu omrežju.

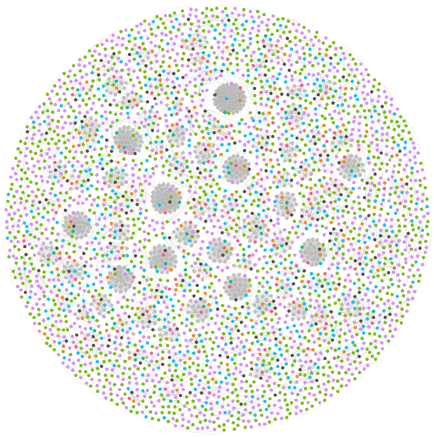
Nato smo se poglobili v posamezne lastnosti vozlišč, ki so bile omenjene v poglavju 2.1.3. Skladno z majhnim premerom omrežja in relativno nizko povprečno stopnjo vozlišč je bil nabor vrednosti za vsako lastnost precej ozek. Cenilka središčnost vmesnosti je bila namreč skoraj vedno enaka nič, uvrstitev strani pa je bil v okviru posameznega grafa vedno enak za vsako



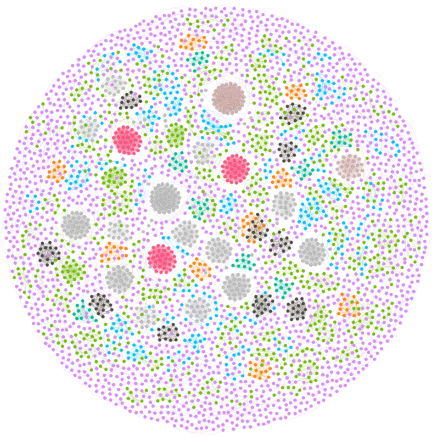
(a) Središčnost vmesnosti



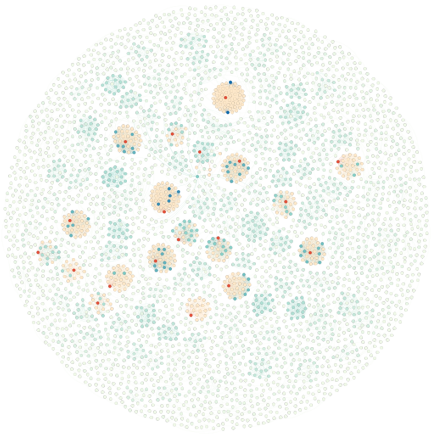
(b) Bližinska središčnost



(c) Središčnost lastnega vektorja



(d) Uvrstitev strani



(e) Stopnja središčnosti

Slika 5.2: Grafični prikaz lastnosti omrežja na osnovi 10 000 primerov. Skupine vozlišč so vidno ločene med seboj.

posamezno vozlišče. Njihove vrednosti so se razlikovale tako malo, da desetmestni decimalni zapis ni bil dovolj, da bi prikazal razlike v izračunu. Kljub temu smo se odločili napovedne algoritme izvesti in del rezultatov je prikazan v nadaljevanju v tabeli 5.11.

Nabor cenilk	MAE	RMSE	R^2	Metoda	Število dreves
AUTH, BC, CC, DC, EVC, PR	1,017	1,212	0,031	RF	100
	1,021	1,220	0,018	LR	/
AUTH, BC, CC, EVC	1,023	1,222	0,014	RF	50
	1,024	1,222	0,014	LR	/
AUTH, BC, DC, EVC, PR	1,017	1,212	0,031	RF	50
	1,024	1,222	0,014	LR	/
DC, PR	1,017	1,212	0,030	RF	10
	1,028	1,228	0,005	LR	/
DC	1,017	1,212	0,031	RF	50
	1,028	1,228	0,005	LR	/

Tabela 5.11: Povprečni rezultati prečnega preverjanja za omrežja iz 10 000 primerov, izvedeno pri 80% učni množici. Vozlišča so posamezni primeri.

Na začetku smo testirali samo lastnosti omrežij, ustvarjenih iz 1000 primerov. Metoda podpornih vektorjev je dala neničelne rezultate le pri treh naborih cenilk. Zato smo se odločili atribute testirati na naslednji večji množici, in sicer na 10 000 primerih. Potrebno je izpostaviti, da metoda podpornih vektorjev s povečevanjem množice postaja slabša, njen koeficient R^2 pa negativen, kar je vidno tudi v tabeli 5.12. To pomeni, da algoritem ni uspešen pri ugotavljanju koreliranosti med izbranimi cenilkami in razredom. Z drugimi besedami, v tem primeru se linearni model metode podpornih vektorjev obnese slabše kot metodo naključnih gozdov. Potrebno je omeniti, da zaradi odstranitve dela vozlišč razredi niso več enako porazdeljeni, kar dodatno

vpliva na slab rezultat.

Nabor cenilk	MAE	RMSE	R^2
AUTH, BC, CC, DC, EVC, PR	1,139	1,361	-0,007
DC	1,142	1,364	-0,011

Tabela 5.12: SVM rezultati pri testiranju 200 000 primerov, pri 80% velikosti učne množice. Algoritem se izkaže slabše kot RF.

To omrežje je konsistentno s prvo in drugo fazo, kjer napovedujemo posamično številčno oceno v podatkih. Vendar se tu pojavi problem v smiselnosti samega omrežja, ker ne dobimo dobro povezanost med vozlišči, kar potrjujejo tudi dobljeni rezultati lastnosti omrežja. Kot je že znano iz tabele 5.10, premer grafov znaša 1, kar pomeni, da gre povprečna najdaljša pot v grafu skozi le eno povezavo. To še dodatno pomeni, da dobimo samo polne podgrafe, kjer vsak podgraf opisuje enega uporabnika in vsaka pot povezuje le dve oceni. Podgrafi so zato nepovezani med seboj.

Takšno modeliranje omrežja ni omogočilo dobre analize lastnosti, zato smo se odločili za rugo zgradbo omrežja, ki sledi v nadaljevanju.

5.3.2 Na osnovi izdelkov

Zaradi neuspešnosti prejšnjega modela, ki je temeljil na istem pristopu kot v prvi in drugi fazi, smo bili motivirani za iskanje morebitne izboljšave in omrežja z boljšo predstavitvijo uporabnikov in primerov. Iz tega je razvidno, da bo pristop določanja napovedovanega razreda sedaj drugačen.

Določili smo, da vozlišča predstavljajo posamezne izdelke oziroma ocenjene filme. Za razred, ki ga želimo napovedati, smo privzeli izračunano povprečje vseh ocen posameznega izdelka, saj je ocen toliko, kolikor je uporabnikov, ki so ta izdelek ocenjevali. To spremeni prvotno strategijo in preko izdelka, ki jih povezuje, združi več primerov v enem. Pri povezavah smo upoštevali enak princip, kot pri prejšnjem modelu: povezava med dvema vo-

zliščema obstaja v primeru, da je vsaj en uporabnik glasoval za oba izdelka.

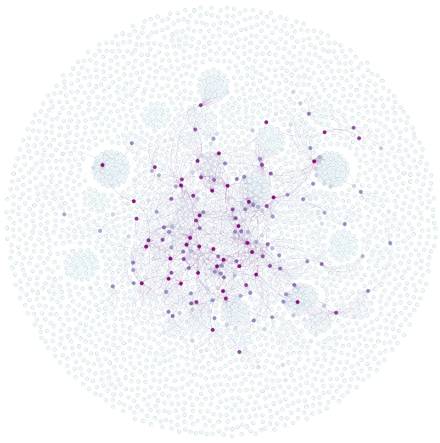
Za primerjavo s prejšnjim omrežjem je podana tabela 5.13. Novo omrežje vsebuje precej manj vozlišč, zaradi česar sta gostota in povezanost grafa večja, kar je razvidno iz povprečne stopnje vozlišča in premera grafa.

Število primerov	Število vozlišč	Število povezav	Povprečna stopnja vozlišča	C	D
1 000	124	113	1,8	0,452	1
10 000	3 212	11 876	7,4	0,607	18
25 000	9 310	72 050	15,5	0,661	12
50 000	18 831	270 178	28,7	0,699	17
100 000	34 516	1 062 720	61,6	0,735	14
150 000	46 788	2 332 721	99,7	0,756	15
200 000	56 959	4 068 648	143,9	0,768	13

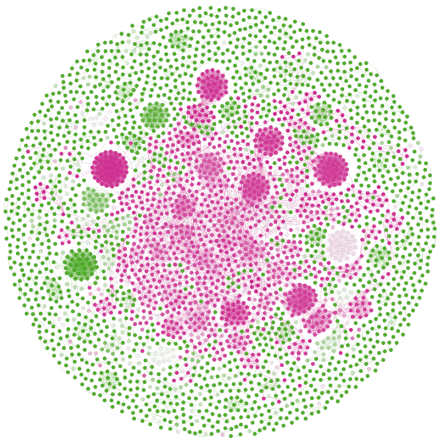
Tabela 5.13: Nekatere lastnosti dobljenih omrežij v eni iteraciji. C označuje nakopičenost, D pa premer omrežja.

V podani tabeli 5.14 so prikazani najboljši rezultati za posamezno kombinacijo cenilk. Zaradi večje preglednosti so vrstice z nesmiselno ničelno oceno koeficienta R^2 izpuščene. Spomnimo se, da nizek R^2 pri rezultatih metode LR kaže na nizko linearno koreliranost med atributi in številčno oceno. Atributa AUTH in HUBS sta identična, ker imamo neusmerjen graf, in zato upoštevamo samo enega.

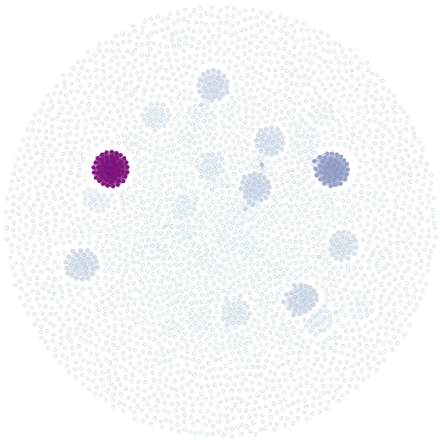
Kljub temu da imamo kombinacijo vseh šestih omrežnih atributov se kot najboljše cenilke izkažejo krajše kombinacije med središčnostmi BC, CC, DC in EVC ter PR, ki izhaja iz EVC. Iz tega je razvidno, da je povezanost filmov na podlagi skupnega glasovanja pomembno. Odstranjevanje pasivnih uporabnikov, ki so glasovali samo enkrat, in napovedovanje povprečne ocene izdelka, poda boljšo oceno kot napovedovanje povprečne vrednosti za vsak primer posebej, kot smo videli v prvi fazi. Iz tega sledi, da je lahko z uporabo le najpreprostejših lastnosti omrežja in preko enostavne zgradbe omrežja moč



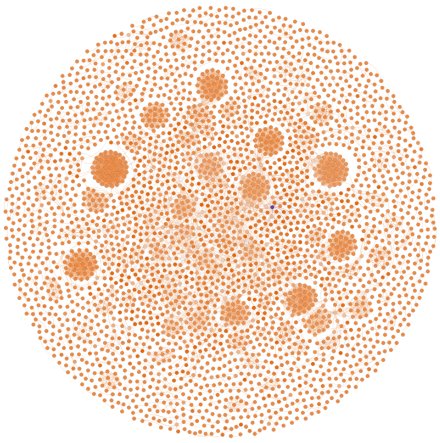
(a) Središčnost vmesnosti



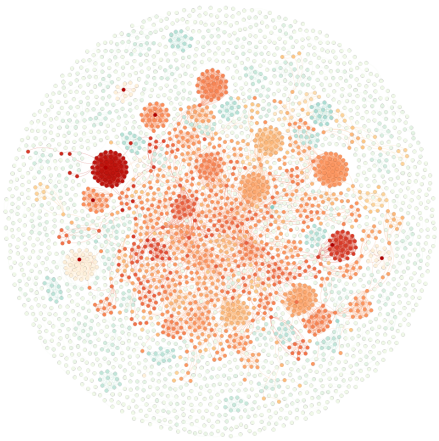
(b) Bližinska središčnost



(c) Središčnost lastnega vektorja



(d) Uvrstitev strani



(e) Stopnja središčnosti

Slika 5.3: Grafični prikaz lastnosti omrežja na osnovi izdelkov, dobljeno iz 10 000 primerov. Skupine vozlišč so bolj povezane med seboj v primerjavi s prejšnjim modelom.

izboljšati rezultat metode srednje vrednosti.

Tabela 5.15 kaže na neuspešnost metode podpornih vektorjev tudi pri maksimalni velikosti množice. Podani rezultati so sicer tudi najboljši rezultati splošno za to metodo pri drugačni zgradbi omrežja. Sicer so skoraj identični najboljšemu rezultatu, dobljenemu preko metode naključnih gozdov, a vidimo, da rezultati spominjajo na prvo fazo. Sklepamo lahko, da je bil SVM najbolj ustrezen za testiranje druge faze, kar pomeni, da se je dobro izkazal pri evaluaciji atributov frekvenčne analize besedil.

Za zadnjo fazo lahko sklepamo, da je najslabši od vseh prvi model omrežja, saj ima graf nizko gostoto in nepovezane podgrafe, kar je bilo vidno tudi iz samih lastnosti omrežja. S povečanjem števila primerov in posledičnim povečanjem števila vozlišč in povezav, dobimo vedno slabše rezultate. Nabor vrednosti vsakega atributa je precej majhen, vrednosti središčnosti pa se z večanjem grafa spreminjajo le malo, vendar to zaradi zaokroževalne napake in natančnosti ni razvidno.

Drugi model omrežja, s sicer drugačnim pristopom napovedovanja, se je izkazal kot le malo boljši. Tu smo napovedovali povprečno oceno filma, ne pa vsake posamezne ocene kot pri prejšnih fazah. Povezanost med filmi na podlagi uporabnikov je dobra osnova za nadaljne izboljšave, vendar bi bil za to potreben naprednejši model omrežja.

Nabor cenilk	MAE	RMSE	R^2	Metoda	Število dreves
AUTH, BC, CC, DC, EVC, PR	0,934	1,160	0,050	RF	10
	0,965	1,180	0,016	LR	/
AUTH, BC, CC, DC, EVC	0,965	1,180	0,016	LR	/
	0,931	1,158	0,053	RF	10
AUTH, BC, DC, EVC, PR	0,944	1,171	0,031	RF	10
	0,966	1,181	0,015	LR	/
BC, CC, DC, EVC	0,924	**1,145**	0,075	RF	100
	0,972	1,188	0,004	LR	/
BC, CC, DC, PR	0,924	**1,144**	0,076	RF	50
	0,971	1,187	0,005	LR	/
BC, CC	0,940	1,163	0,045	RF	100
	0,970	1,188	0,003	LR	/
BC	0,973	1,190	0,000	LR	/
CC	0,956	1,176	0,023	RF	100
	0,972	1,188	0,003	LR	/
DC, PR	0,955	1,184	0,010	RF	10
	0,972	1,188	0,004	LR	/
DC	0,971	1,176	0,023	RF	50
	0,974	1,190	0,000	LR	/
PR	0,973	1,190	0,000	LR	/

Tabela 5.14: Povprečni rezultati prečnega preverjanja za graf iz 10 000 primerov, dobljeni z učenjem na 80% učni množici. Vozlišča so posamezni filmi oziroma izdelki.

Nabor cenilk	MAE	RMSE	R^2
AUTH, BC, CC, DC, EVC, PR AUTH, BC, CC, DC, EVC AUTH, BC, DC, EVC, PR BC, CC, DC, EVC	0,927	1,148	0

Tabela 5.15: SVM za 200 000 primerov. Rezultati so dobljeni pri učenju na 80% velikosti učne množice. Ocene so enake za vse kombinacije, kar kaže na nespešnost modela.

Poglavje 6

Sklepne ugotovitve

Namen naloge je bil na podlagi treh različnih pristopov podatkovnega rudarjenja in strojnega učenja najti in medsebojno primerjati izbiro cenilk na konkretnem primeru. Na podlagi rezultatov in sklepanj, predstavljenih v prejšnjem poglavju, smo dobili vpogled v izbiro atributov za napovedovanje spletnih ocen. Obenem smo se poglobili v novo programsko okolje in se lotili dela z večjo množico podatkov, kar je zahtevalo nekoliko drugačen pristop v primerjavi z dosedanjimi izkušnjami.

V prvi fazi se je izkazalo, da primitivna obdelava časovnih atributov in nabor cenilk, ki se preko njih izgradi, ni dobra izbira cenilk, ki bi nam omogočala uspešne napovedi ocen. To nam lahko k izboljšanju napovedi pomaga le v kombinaciji z ostalimi atributi.

Pričakovanja glede tretje faze so bila delno izpolnjena. S prvim modelom omrežja, ki je bil konsistenten s prvo in drugo fazo glede napovedovanega razreda, smo dobili najslabše rezultate. Razlog za to je sama zgradba omrežij oziroma grafov, ki z nizko gostoto in atributi vozlišč, ki jih dobimo, ne zadoščajo za izvedbo dobre napovedi. Od tod izhaja ideja o drugačni zgradbi omrežja na podlagi drugih kriterijev. Kot najbolj osnovno in drugačno verzijo omrežja smo zgradili model, omenjen v poglavju 5.3.2, kjer smo dobili

graf z boljšimi lastnostmi in posledično boljše napovedi. Drugi model ne moremo v celoti primerjati s prejšnjimi rezultati, saj smo napovedovali nekoliko drugačno dobljen razred.

Iz prikazanih rezultatov lahko zaključimo, da najboljši nabor cenilk predstavljajo besede komentarja, ki sledijo vsaki oceni v posameznem primeru. Kljub temu da je bila uporabljena le frekvenčna analiza besed, smo v tem primeru dobili najvišjo točnost napovedi. Iz tega lahko sklepamo, da se je druga faza s področja rudarjenja besedil izkazala kot obetavna tako za nadaljno obdelavo besedilnega korpusa kot tudi za uvedbo kompleksnejših metod analize. Uspešnost druge faze dodatno potrjuje, da so uporabniki oziroma komentatorji na spletu živ in dinamičen sistem ter da lahko njihov govor ogromno pove o njihovem obnašanju in razmišljanju na spletu. Takšne interakcije bi lahko predstavili preko kombiniranja druge in tretje faze, kjer bi poudarili podobnosti komentarjev preko zgradbe omrežja.

V nalogi se nismo lotili napredne kronološke analize primerov. To bi zahtevalo uporabo zapletenejših pristopov in celo izdelavo priporočilnega sistema, kar je posebno in kompleksno področje v podatkovnem rudarjenju.

Dodatek A

Programska koda

V nadaljevanju sledijo nekaj tipičnih primerov uporabe programskih orodij za namen naloge.
Primer branja podatkovne strukture DataFrame, pridobitev časovnih cenilk in zapisovanje v nov DataFrame:

```
def write_file_timestamp_details(file_name, year_to_test=2010):
    df = pandas.read_csv(file_name, sep='\t', header=0)
    users_timestamp_attr = []

    for index, row in df.iterrows():
        #check if there is a timestamp in the review
        if 'time' in row.keys():
            review = []
            review.append(row['productId'])
            review.append(row['userId'])
            review.append(row['helpfulness'])

            #Extract the separate attributes from the timestamp
            review.append(time.gmtime(int(row['time'])).tm_sec)
            review.append(time.gmtime(int(row['time'])).tm_min)
            review.append(time.gmtime(int(row['time'])).tm_hour)
            review.append(time.gmtime(int(row['time'])).tm_wday)
```

```
            review.append(time.gmtime(int(row['time'])).tm_mday)
            review.append(time.gmtime(int(row['time'])).tm_yday)
            review.append(time.gmtime(int(row['time'])).tm_mon)

            #Add the classes for prediction
            review.append(row['score'])

            #Add the example to the data set
            users_timestamp_attr.append(review)

        else:
            continue

    #Save the attributes in a DataFrame
    df_all = pandas.DataFrame(
      {'productId': [row[0] for row in users_timestamp_attr],
       'userId': [row[1] for row in users_timestamp_attr],
       'second': [row[2] for row in users_timestamp_attr],
       'minute': [row[3] for row in users_timestamp_attr],
       'hour': [row[4] for row in users_timestamp_attr],
       'dayWeek': [row[5] for row in users_timestamp_attr],
       'dayMonth': [row[6] for row in users_timestamp_attr],
       'dayYear': [row[7] for row in users_timestamp_attr],
       'month': [row[8] for row in users_timestamp_attr],
       'score': [row[9] for row in users_timestamp_attr],
       'helpfulness': [row[10] for row in users_timestamp_attr]})

    #Export the DataFrame to a file
    df_filename_all = 'non_text_timestamp_attr_%d.csv' % year_to_test
    df_all.to_csv(df_filename_all, sep='\t', index=False)
```

Izvoz podatkovne strukture DataFrame v LaTeX obliki:

```
#Choose the results for 40000x5 or 200000 examples
#and only one test set size and learner
df_temp = df[df["n_examples"] == 40000]
df_temp = df_temp[df_temp["train_set_size"] == 0.8]
df_temp = df_temp[df_temp["learner"] == "SVM"]

#Drop the unnecessary columns and sort by the column "features"
df_temp = df_temp.drop(["AVG MSE", "n_examples", "trees",
    "learner", "train_set_size"], axis=1,
    inplace=False).sort("features")

#Choose the order in which the columns would print
df_temp = df_temp[["features", "AVG MAE" , "AVG RMSE", "AVG R2"]]

#Export to LaTeX format
df_latex = df_temp.to_latex(index=False)
print df_latex
```

Pridobitev frekvenc tfidf:

```
def analyze_text(text_data, mf=None):
    vectorizer = TfidfVectorizer(decode_error='ignore',
        analyzer='word', max_df=0.3, max_features=mf)
    tfidf = vectorizer.fit_transform(text_data.ravel())

    #Export the weights in a DataFrame
    df_tfidf = pandas.DataFrame(tfidf,
        columns=vectorizer.get_feature_names())

    #Dictionary containing the words and their weights
    dict_weights = dict(zip(vectorizer.get_feature_names(),
        tfidf.data))
```

Primer uporabe regresijskega algoritma naključnih gozdov:

```
def rndforest_classifier_test(X, y, test_size, cross_val_param,
    n_trees, n_e, mf=None):
    #Build the model:
    regr = ensemble.RandomForestRegressor(n_estimators=n_trees,
        n_jobs=-1, random_state=0)
    f_name_test = 0

    for t in test_size:
        #Divide the sets into training and into testing set:
        X_train, X_test, y_train, y_test =
            cross_validation.train_test_split(X, y, test_size=t,
            random_state=0)

        #Cross validation on the train set
        for c in cross_val_param:
            for metric in reg_metrics:
                #Calculate MAE, RMSE and R2
                scores_rf = cross_validation.cross_val_score(regr,
                    X_train, y_train, cv=c,
                    scoring=make_scorer(metric))

                #Print MAE, RMSE and R2
                print_cv_error_report()

    #Test the data on the held-out set:
        #Train the data
        regr.fit(X_train, y_train)
        #Test the data
        y_pred = regr.predict(X_test)

        #Print test set errors:
        print_error_report(y_test, y_pred, t)
```

Primer funkcije pri drugem modelu omrežja. Združuje podatke o vozlišč in *productId*, ter izračuna povprečne ocene:

```
def second_join_features_scores(file_name_scores_df,
    file_name_features, file_name_product_ids):
    df_scores = pd.read_csv(file_name_scores_df, sep='\t')
    df_feat = pd.read_csv(file_name_features, sep='\t')
    df_product_ids = pd.read_csv(file_name_product_ids, sep='\t')

    products = list(set(df_product_ids["productId"].values))
    scores_by_product = dict()

    #Assign the scores to their products (there are more scores to
        one product)
    for index, row in df_scores.iterrows():
        if row["productId"] in products: #If the product belongs to
            the graph, add the score
            if not scores_by_product.has_key(row["productId"]):
                scores_by_product[row["productId"]] = [row["score"]]
            else:
                scores_by_product[row["productId"]].append(row["score"])

    #Calculate average for the scores of each product
    scores_by_product_average = dict()
    for product in scores_by_product:
        scores_by_product_average[product] =
            np.mean(scores_by_product[product])

    df_class = pd.DataFrame({'productId': [product for product in
        scores_by_product_average],
                             'score':
                                 [scores_by_product_average[product]
                                 for product in
                                 scores_by_product_average]})
```

```
    #Join the DataFrames into one
    merged_df1 = pd.merge(left=df_product_ids, right=df_class,
        left_on='productId', right_on='productId')
    merged_df2 = pd.merge(left=merged_df1, right=df_feat,
        left_on='index', right_on='NodeID')

    #Delete 'index' columns since it's the same as 'NodeID'
    main_merged_df = merged_df2.drop("index", axis=1)
    #Rename the columns for LaTeX output
    main_merged_df = main_merged_df.rename(index=str, columns =
        {"Authorities": "AUTH", "BetweennessCentrality":"BC",
        "ClosenessCentrality":"CC", "DegreeCentrality":"DC",
        "EigenVectorCentrality":"EC", "Hubs":"HUBS",
        "PageRank":"PR"})

    return main_merged_df
```

Primer funkcij pri drugem modelu omrežja. Prva sestavi seznam vozlišč in seznam povezav, ki sta potem v drugi funkciji združene v graf:

```
def create_nodes_and_edges(df):
  nodes = []
  edges = []
  data = {}
  data_by_userId = dict()

  productID_index = {}
  #Get all the different values for productId
  products = list(set(df["productId"].values))
  n_nodes = len(products)

  #Rewrite all the products as integers, so Snap.py can assemble a
      graph
  for node_index, product_id in zip(range(n_nodes), products):
      productID_index[product_id] = node_index
  nodes = productID_index.values()

  for index, row in df.iterrows():
      data[index] = [row["score"], row["userId"], row["productId"]]

      #Assign the product to a user
      if not data_by_userId.has_key(row["userId"]):
          data_by_userId[row["userId"]] = [row["productId"]]
      else:
          data_by_userId[row["userId"]].append(row["productId"])

  #Assemble the edge list
  for entry in data_by_userId:
      #User has voted for more than one product:
      if len(data_by_userId[entry]) > 1:
```

```
            for index1, index2 in
                itertools.combinations(data_by_userId[entry], 2):
                edges.append((productID_index[index1],
                    productID_index[index2]))

    return nodes, edges, data, productID_index


def create_graph(nodes, edges, g_name):
    Graph = snap.TUNGraph.New() #Create an undirected empty graph

    #We add the nodes and edges to an empty graph
    for nodeId in nodes:
        Graph.AddNode(nodeId)
    for first, second in edges:
        Graph.AddEdge(first, second)
    print "Stevilo vozlisc pred brisanju: %d" % Graph.GetNodes()

    snap.DelZeroDegNodes(Graph) #Delete the non-connected nodes
    print "Stevilo vozlisc po brisanju: %d" % Graph.GetNodes()

    snap.SaveEdgeList(Graph, g_name, "Tab-separated list of edges")
```

Seznam tabel

Seznam slik

Literatura

[1] Grafični vmesnik: Gephi. https://gephi.org/. [Dostopano 3. 9. 2016].

[2] Programska knjižnica: Numpy. http://www.numpy.org/. [Dostopano 2. 9. 2016].

[3] Programska knjižnica: pandas. http://pandas.pydata.org/. [Dostopano 2. 9. 2016].

[4] Programska knjižnica: scikit-learn. http://scikit-learn.org/stable/. [Dostopano 2. 9. 2016].

[5] Programska knjižnica: Snap.py. http://snap.stanford.edu/snappy/. [Dostopano 3. 9. 2016].

[6] Shema hiperravnin pri metodi podpornih vektorjev. https://upload.wikimedia.org/wikipedia/commons/2/2a/Svm_max_sep_hyperplane_with_margin.png.

[7] Web data: Amazon movie reviews. http://snap.stanford.edu/data/web-Movies.html.

[8] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: An open source software for exploring and manipulating networks. 2009.

[9] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[10] Allan Borodin, Gareth O Roberts, Jeffrey S Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *Proceedings of the 10th international conference on World Wide Web*, pages 415–429. ACM, 2001.

[11] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[12] Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[13] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[14] Nello Cristianini and John Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[15] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[16] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[17] Tin Kam Ho. Random decision forests. 14-16 Avgust 1995.

[18] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[19] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

[20] Igor Kononenko and Matjaz Kukar. *Machine Learning and Data Mining: Introduction to Principles and Algorithms*. Horwood Publishing Limited, 2007.

[21] Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1, 2016.

[22] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*, volume 999. MIT Press, 1999.

[23] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908. ACM, 2013.

[24] Wes McKinney. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. "O'Reilly Media, Inc.", 2012.

[25] John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Ehud Reiter, Robert Dale, and Zhiwei Feng. *Building natural language generation systems*, volume 33. MIT Press, 2000.

[29] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2005.